NLP – Unit 2 (Language Syntax and Semantics) – IN-SEM PYQ Answers

Q1. Define Morphology? Explain stem and affix classes of morphemes with Example. [5]

Morphology is the branch of linguistics that studies the **internal structure of words** and the formation of words using the smallest meaningful units called **morphemes**.

A **morpheme** is the smallest unit of meaning in a language.

Classes of Morphemes

Morphemes are mainly classified into:

1. **Stem (Root Morpheme)**
    - The core part of a word that carries the principal lexical meaning.
    - It can stand alone as a word (free morpheme).
    - It forms the base to which affixes are attached.
    - Examples:
        i. "play" in playing
        ii. "happy" in happiness
        iii. "write" in rewriting
    - In "playing" →
      Stem = play
2. **Affix (Bound Morpheme)**
    - A morpheme that cannot stand alone.
    - It is attached to a stem to modify its meaning or grammatical function.
    - Types of Affixes:
      a) **Prefix** (attached before stem)
      Example:
      unhappy → un + happy
      b) **Suffix** (attached after stem)
      Example:
      playing → play + ing
      teacher → teach + er
      c) (Less common) Infix, Circumfix (not common in English)

Q2. What is morphology? Explain Derivational and Inflectional morphology in detail with suitable examples. [7]

Q3. Explain Derivational and Inflectional morphology in detail with suitable Example. [8]

1. Derivational Morphology

Derivational morphology deals with the formation of **new words** by adding affixes to a root or stem. It often changes the **meaning** and sometimes the **part of speech (POS)** of the word.

Characteristics:

- Creates new lexical entries in the dictionary.

- May change grammatical category.
- Not mandatory for grammatical correctness.

Examples:

1. happy (Adjective) → happiness (Noun)
   happy + ness
2. teach (Verb) → teacher (Noun)
   teach + er
3. possible (Adjective) → impossible (Adjective, meaning changed)
   im + possible
4. nation (Noun) → national (Adjective) → nationalize (Verb)

Thus, derivational morphology contributes to vocabulary expansion.

2. Inflectional Morphology

Inflectional morphology modifies a word to express **grammatical information** such as tense, number, gender, case, degree, etc., without changing its core meaning or part of speech.

Characteristics:

- Does not create new words.
- Does not change word class.
- Required by grammatical rules.
- Occurs after derivational affixes (if present).

Examples:

1. play → plays → played → playing
   (tense/aspect changes)
2. book → books
   (singular to plural)
3. tall → taller → tallest
   (degree of comparison)
4. boy → boy's
   (possessive case)

Here, the base meaning remains the same, only grammatical function changes.

Comparison

- Derivational → Changes meaning and sometimes POS.
- Inflectional → Expresses grammatical relations without changing POS.

Q4. Illustrate the working of FST for morphological analysis with an example. [7]

A **Finite State Transducer (FST)** is a finite automaton with input and output labels.
It is used in NLP to map between:

Surface form ↔ Lexical form

In morphological analysis, FST decomposes a word into:

Root + Morphological features (suffix/prefix tags)

Working of FST in Morphological Analysis

An FST consists of:

- Finite set of states
- Input symbols (surface word characters)
- Output symbols (lexical representation)
- Transitions with input/output pairs

It scans the input word character by character and produces the corresponding lexical form.

Example: Analysis of the word "cats"

Goal:
cats → cat + Noun + Plural

Step-wise Working:

1. Read 'c' → output 'c'
2. Read 'a' → output 'a'
3. Read 't' → output 't'
4. Read 's' → output "+N +PL"

Thus,

Input: cats
Output: cat + N + PL

State Diagram of FST

```
     c/c       a/a       t/t
(q0) ------> (q1) ------> (q2) ------> (q3)
                                |
                              s/+N+PL
                                |
                              (q4)
```

Explanation:

- q0 → q3 builds the stem "cat"
- On reading 's', transition outputs morphological tag "+N +PL"
- q4 is final accepting state

Another Example: "playing"

playing → play + V + PROG

The FST:

- Recognizes stem "play"
- On reading "ing" outputs "+V +PROG"

Importance in NLP

- Efficient morphological parsing
- Used in lexical analyzers
- Handles regular inflectional morphology
- Bidirectional (can perform analysis and generation)

Q5. Explain CFG with suitable example. [5]

A **Context Free Grammar (CFG)** is a formal grammar used to describe the syntactic structure of natural languages.
It consists of production rules where a single non-terminal symbol is replaced by a sequence of terminals and/or non-terminals.

A CFG is defined as a 4-tuple:

G = (N, T, P, S)

Where:

- N → Set of Non-terminals
- T → Set of Terminals
- P → Set of Production Rules
- S → Start Symbol

Properties of CFG

- Left-hand side of every rule contains only one non-terminal.
- Suitable for representing hierarchical structure of sentences.
- Widely used in syntactic parsing.
- Forms the basis of Probabilistic CFG (PCFG).

Example

Consider the sentence:
"the boy eats an apple"

Grammar:

S → NP VP
NP → Det N
VP → V NP
Det → the | an
N → boy | apple
V → eats

Derivation

S
→ NP VP
→ Det N VP
→ the boy VP
→ the boy V NP
→ the boy eats NP
→ the boy eats Det N
→ the boy eats an apple

Q6. What is Probabilistic context-free grammars? State the benefits of probabilistic parsing. [7]

A **Probabilistic Context-Free Grammar (PCFG)** is an extension of Context-Free Grammar (CFG) in which each production rule is associated with a **probability**.

In PCFG, for a non-terminal AAA, the probabilities of all rules expanding AAA sum to 1:

$$\sum P(A \rightarrow \beta)=1$$

A PCFG is defined as:

G = (N, T, P, S, Prob)

Where:

- N → Non-terminals
- T → Terminals
- P → Production rules
- S → Start symbol
- Prob → Probability of each rule

The probability of a parse tree is the **product of probabilities** of all production rules used in that derivation.

Example

Grammar:

S → NP VP (1.0)
NP → Det N (0.6)
NP → NP PP (0.4)
VP → V NP (0.7)
VP → VP PP (0.3)

Sentence: “astronomers saw stars with ears”

Because of PP-attachment ambiguity, multiple parse trees are possible.
PCFG assigns probabilities to each parse and selects the **most probable parse tree**.

Benefits of Probabilistic Parsing

1. **Handles Ambiguity:** Chooses the most likely parse among multiple syntactically valid parses.
2. **Data-Driven:** Rule probabilities are learned from annotated corpora (Treebanks).
3. **Improves Accuracy:** Reduces incorrect parses compared to pure CFG.
4. **Captures Linguistic Preferences:** Reflects real-world usage frequency of grammatical constructions.
5. **Foundation for Statistical Parsing:** Forms basis for probabilistic and statistical NLP models.

Q7. Construct the possible parse trees for the sentence “astronomers saw the stars with ears” and comment on the ambiguity associated with it. (Assume suitable grammar).[8]

Sentence: “astronomers saw the stars with ears”

Assumption: Context Free Grammar (CFG) is assumed.

Example Grammar:

S → NP VP
VP → V NP | VP PP
NP → N | Det N | NP PP
PP → P NP
Det → the
N → astronomers | stars | ears
V → saw
P → with

The sentence is structurally ambiguous because the Prepositional Phrase (PP) “with ears” can attach either to:

1. The Verb Phrase (VP) → modifies “saw”
2. The Noun Phrase (NP) → modifies “stars”

Parse Tree 1: PP attaches to VP
(Meaning: Astronomers used ears to see the stars)

```
         S
        /  \
       NP    VP
       |    /  \
 astronomers   VP    PP
             / \   / \
            V  NP   P  NP
           saw |    |   |
              Det N with ears
               |   |
              the stars
```

Structure:
S → NP (astronomers) + VP
VP → VP + PP
PP modifies the verb “saw”

Parse Tree 2: PP attaches to NP
(Meaning: The stars have ears)

```
            S
          /  \
        NP     VP
        |    /  \
  astronomers   V     NP
            saw   /  \
                NP    PP
               /\   /\
              Det  N  P  NP
              |    |  |   |
             the stars with ears
```

Structure:
S → NP + VP
VP → V + NP
NP → NP + PP
PP modifies "stars"

Ambiguity Explanation

This is a case of **structural (syntactic) ambiguity** caused by **Prepositional Phrase attachment**.

- In Tree 1, "with ears" modifies the action (instrument interpretation).
- In Tree 2, "with ears" modifies the noun "stars" (descriptive interpretation).

Since the grammar allows both NP → NP PP and VP → VP PP, two valid parse trees are generated.

Q8. Derive a top-down, depth-first, left-to-right parse tree for the given sentence [7] :

"The angry bear chased the frightened little squirrel "

Use the following grammar rules to create the parse tree:

S -> NP VP	Det -> the
NP -> Det Nom	Adj -> little | angry | frightened
VP -> V NP	N -> squirrel | bear
Nom -> Adj Nom | N	V -> chased

Given Sentence: "The angry bear chased the frightened little squirrel"

Grammar Rules:

S → NP VP
NP → Det Nom
Nom → Adj Nom | N
VP → V NP
Det → the
Adj → little | angry | frightened
N → squirrel | bear
V → chased

Top-Down, Depth-First, Left-to-Right Derivation

1. Start with start symbol
   S
2. Apply S → NP VP
   NP VP
3. Expand NP → Det Nom
   Det Nom VP
4. Det → the
   the Nom VP
5. Nom → Adj Nom
   the Adj Nom VP
6. Adj → angry
   the angry Nom VP
7. Nom → N
   the angry N VP
8. N → bear
   the angry bear VP
9. Expand VP → V NP
   the angry bear V NP
10. V → chased
    the angry bear chased NP
11. Expand NP → Det Nom
    the angry bear chased Det Nom
12. Det → the
    the angry bear chased the Nom
13. Nom → Adj Nom
    the angry bear chased the Adj Nom
14. Adj → frightened
    the angry bear chased the frightened Nom
15. Nom → Adj Nom
    the angry bear chased the frightened Adj Nom
16. Adj → little
    the angry bear chased the frightened little Nom
17. Nom → N
    the angry bear chased the frightened little N
18. N → squirrel

Final Derived Sentence: the angry bear chased the frightened little squirrel

Parse Tree (Top-Down, Depth-First, Left-to-Right)

```
              S
            /   \
          NP     VP
         / \    / \
       Det   Nom  V    NP
       |    / \ |   / \
      the  Adj  Nom chased  Det   Nom
         |    |        |   /   \
        angry  N        the  Adj  Nom
               |         |   / \
              bear      frightened Adj  Nom
                           |     |
                          little   N
                                 |
                              squirrel
```

Q9. Explain with suitable examples following relationship between word meanings [8] :
1. Homonymy 2. Polysemy 3. Synonymy 4. Hyponymy 5. Antonymy 6. Hypernomy 7. Meronomy

1. Homonymy

- Homonymy refers to words having **same form (spelling/pronunciation)** but **completely unrelated meanings**.
- The meanings have **no semantic connection**.
- Can be classified as **homophones** (same sound) and **homographs** (same spelling).
- Causes **lexical ambiguity** in NLP.
- Requires **Word Sense Disambiguation (WSD)** for correct interpretation.

  Examples:

  - bank → financial institution / river bank
  - bat → flying mammal / cricket bat

2. Polysemy

- Polysemy refers to a single word having **multiple related meanings**.
- Meanings are **semantically connected**.
- Very common in natural languages.
- Context determines the intended sense.
- Important in semantic analysis and WSD.

  Examples:

  - head → body part / leader / top of table
  - paper → material / research article

3. Synonymy

- Synonymy is a relationship between words with **similar meanings**.
- Complete synonymy is rare; usually there are **slight contextual differences**.
- Synonyms may differ in **formality, usage, or intensity**.
- Useful in text generation and information retrieval.
- Helps in query expansion in NLP.

  Examples:

  - big – large
  - start – begin
  - happy – joyful

4. Hyponymy

- Hyponymy represents a **"type-of" relationship**.
- A hyponym is a **more specific term** under a general category.
- Establishes hierarchical semantic relations.
- Used in ontology and taxonomy building.
- Important in lexical databases like WordNet.

  Examples:

  - rose → hyponym of flower
  - dog → hyponym of animal
  - car → hyponym of vehicle

5. Antonymy

- Antonymy refers to words having **opposite meanings**.
- Types: **Gradable, Complementary, Relational**.
- Gradable antonyms allow degrees (hot–cold).
- Complementary antonyms are binary (alive–dead).
- Important in sentiment analysis.

  Examples:

  - hot – cold
  - buy – sell
  - happy – sad

6. Hypernymy (Hypernomy)

- Hypernym is a **general or broader term**.
- It represents a higher level in semantic hierarchy.
- Opposite of hyponym.
- Used in semantic networks and ontology construction.
- Helps in concept generalization.

  Examples:

  - animal → hypernym of dog

- fruit → hypernym of apple
- vehicle → hypernym of car

7. Meronymy

- Meronymy represents a **“part-of” relationship**.
- Indicates component–whole relationship.
- Opposite relation is **holonymy** (whole–part).
- Important in knowledge representation.
- Used in semantic networks like WordNet.

Examples:

- wheel → part of car
- chapter → part of book
- finger → part of hand

Q10. What are different techniques for the semantic analysis for the Statement.

Different **techniques for semantic analysis of a statement (in NLP)** are used to determine the **meaning of words, phrases, and sentences**. Common techniques include:

1. **Lexical Semantics**
    - Studies the meaning of individual words and their relationships.
    - Includes relations like **synonymy, antonymy, hyponymy, and polysemy**.
2. **Dictionary-Based Approach:** Uses lexical databases or dictionaries such as WordNet to determine word meanings and relationships.
3. **Word Sense Disambiguation (WSD):** Determines the correct meaning of a word when it has multiple senses depending on context.
4. **Latent Semantic Analysis (LSA):** A statistical technique that analyzes relationships between words and documents by identifying hidden semantic structures in text.
5. **Semantic Networks:** Represents knowledge using a network of concepts connected by semantic relationships.
6. **Frame Semantics / Case Grammar:** Analyzes meaning based on roles in a sentence (e.g., agent, action, object).
7. **Ontology-Based Semantics:** Uses structured knowledge bases or ontologies to define relationships between concepts.
8. **Distributional Semantics:** Determines meaning based on how words appear in similar contexts across large text corpora.
9. **Compositional Semantics:** Builds the meaning of a full statement from the meanings of its individual words and their grammatical structure.

★ ***PLEASE VERIFY THE NUMERICALS*** ★